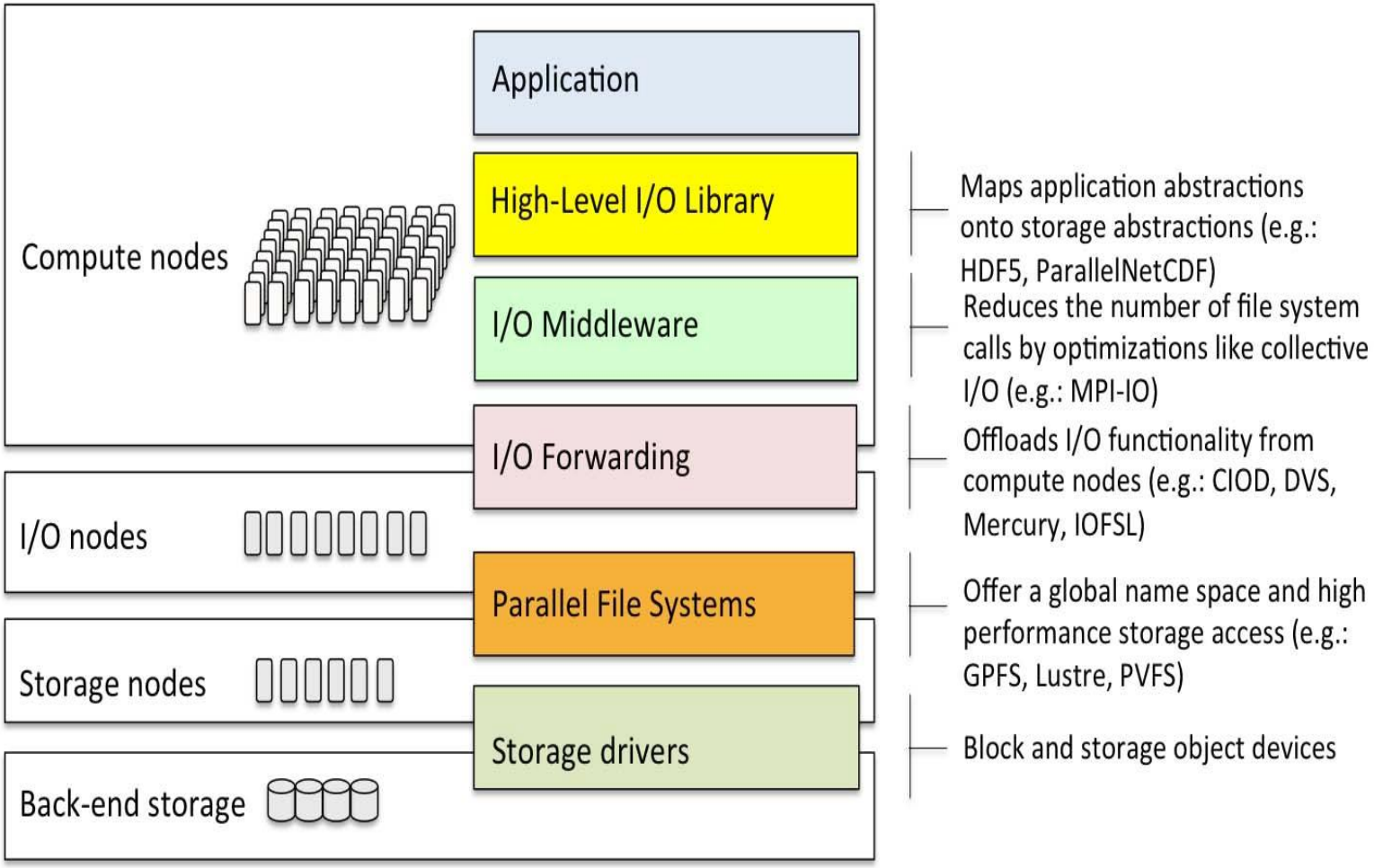
Network coded Storage I/O subsystem for HPC exascale applications

INTRODUCTION

- Intel predicts 10 times (4.4ZB to 44 ZB) data explosion between 2013-2020.
- Massive data explosion makes legacy storage management complex and inefficient.
- The research industry demands technology convergence of Bigdata and HPC to handle application demands.

HPC I/O STACK


Compute nodes
Application
High-Level I/O Library
I/O Middleware
I/O Forwarding
I/O nodes
Parallel File Systems
Storage nodes
Storage drivers
Back-end storage
Maps application abstractions onto storage abstractions (e.g.: HDF5, ParallelNetCDF)
Reduces the number of file system calls by optimizations like collective I/O (e.g.: MPI-IO)
Offloads I/O functionality from compute nodes (e.g.: CIOD, DVS, Mercury, IOFSL)
Offer a global name space and high performance storage access (e.g.: GPFS, Lustre, PVFS)
Block and storage object devices

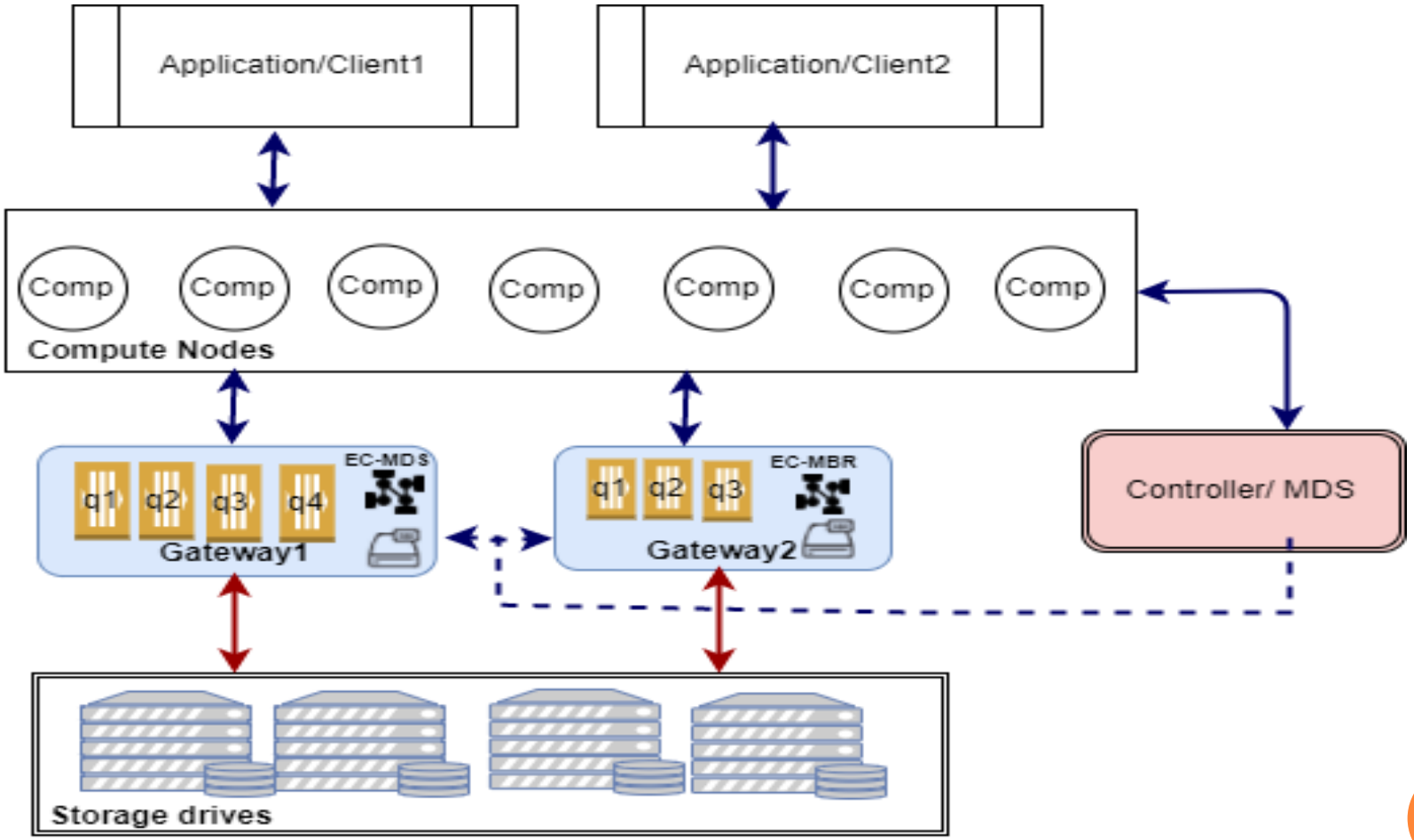
I/O Stack Software challenges for Exascale

- Growth in number of storage devices and application-
  - Computer science challenge
  - Data placement and handling challenge
  - Fault tolerance provisioning
  - Data model with high productivity interface

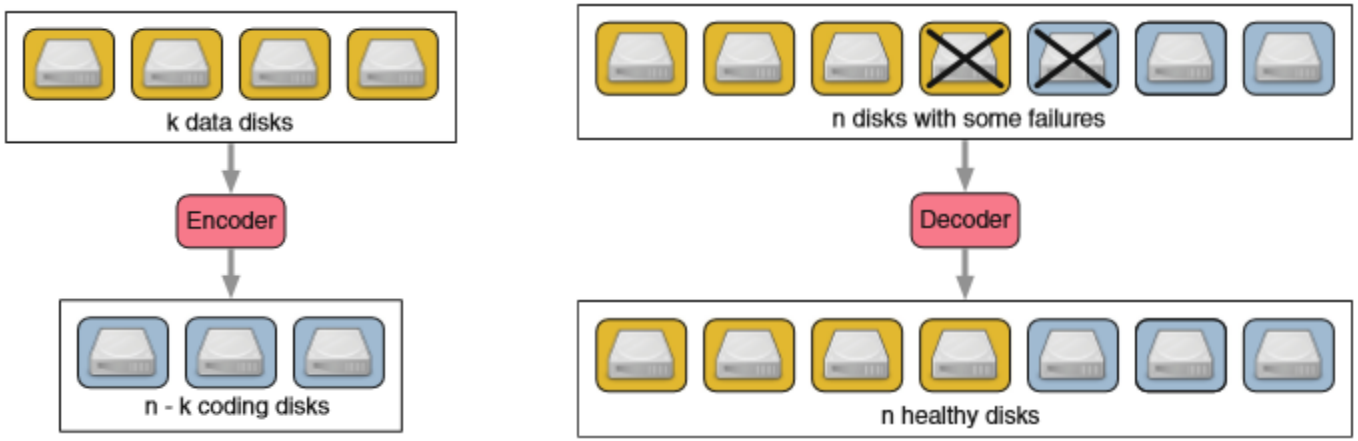
Proposed approach

- Storage I/O Subsystem with following components
  - Software defined storage gateway architecture
    - Storage gateway dataplane
    - Control plane coexist with MDS
  - Caching techniques with hierarchical storage structure
    - Embedded SSD storage at storage gateway
  - Multi-tier (multi-level) network coding scheme
    - At storage gateway with preconfigured different erasure coding scheme
    - Erasure code based on systematic RS code (n,k)

System architecture

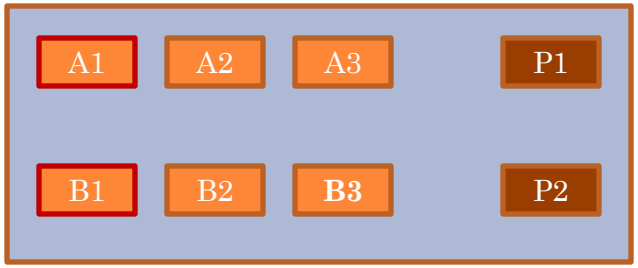


Erasure Coding details



Erasure Coding encoding and decoding

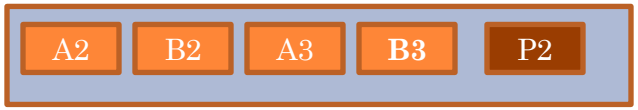
(3,1)RS code



One tier Erasure code

(2,1)RS code

(4,1)RS code



Two tier Erasure code

Open source erasure coding libraries

Sr. No	Library name	Characteristics	Languages suported
1	Zfec	RS Encoding, Vandermonde matrix	C, Python, Haskell
2	Liberasure code	RS Encoding, Vandermonde matrix, XOR	C, Python,
3	Jersure	RS Encoding, Vandermonde matrix, Cauchy RS	C, Python,
4	ISA-L	RS Encoding, Vandermonde matrix,	C, Python,
5	gflib	RS Encoding, Vandermonde matrix,	C
6	Backblaze	RS Encoding, Vandermonde matrix,	Java

I/O Access Procedure

- File layout information locates at MDS (Metadata Server) /Controller.
- Client sends an inquire request(read or write) to MDS get file location information before issuing actual I/O requests.
- First inquire packet will carry application-bonded demands with detailed parameters (I/O size, storage overhead, fault tolerance)
- Controller in MDS acknowledges with explicit gateway nodes and storage nodes that will serve the application.

I/O Access Procedure

- Then initial data plane rules will be built in those gateway nodes according to application demand.
- MDS will return file metadata and gateway message to the client node with subsequent routing information.
- Clients then issues actual I/O request directly to the gateway bypassing MDS/controller.
- Gateway processes them based on the pre-built rules.
  - Type of I/O-(request to be assigned to SSD/ HDD)
  - Fault tolerance , storage overhead – build erasure coded scheme on data.

Important References

- R. Springmeyer, C. Still, M. Schulz, J. Ahrens, S. Hemmert, R. Minnich, P. McCormick, L. Ward, D. Knoll, “From Petascale to Exascale: Eight Focus Areas of R&D Challenges for HPC Simulation Environments”
- Jason Hick-NERSC, Storage System Group Lead- LBNL, “I/O Requirements for Exascale”, 2011
- Thereska, Eno, et al. ”Ioflow: A software-defined storage architecture.” Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, 2013
- Wei Shi et al. DEFIO: A Software Defined Storage Network Architecture in HPC Environments at 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS)
- Dorian Burihabwa, Pascal Felber, Hugues Mercier, and Valerio Schiavoni “A Performance Evaluation of Erasure Coding Libraries for Cloud-Based Data Stores (Practical Experience Report)”, IFIP 2016
- Jun Li, Baochun Li , “Zebra: Demand-aware erasure coding for distributed storage systems”, Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium
- Florin Isaila, Jesus Carretero and Rob Ross, “CLARISSE: a middleware for data-staging coordination and control on large-scale HPC platforms”, 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing.